

[0005] A CDMA system may be designed to support one or more CDMA standards such as (1) the "TIA/EIA-95-B Mobile Station-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System" (the IS-95 standard), (2) the "TIA/EIA-98-C Recommended Minimum Standard for Dual-Mode Wideband Spread Spectrum Cellular Mobile Station" (the IS-98 standard), (3) the standard offered by a consortium named "3rd Generation Partnership Project" (3GPP) and embodied in a set of documents including Document Nos. 3G TS 25.211, 3G TS 25.212, 3G TS 25.213, and 3G TS 25.214 (the W-CDMA standard), (4) the standard offered by a consortium named "3rd Generation Partnership Project 2" (3GPP2) and embodied in a set of documents including "TR-45.5 Physical Layer Standard for cdma2000 Spread Spectrum Systems," the "C.S0005-A Upper Layer (Layer 3) Signaling Standard for cdma2000 Spread Spectrum Systems," and the "C.S0024 cdma2000 High Rate Packet Data Air Interface Specification" (the cdma2000 standard), and (5) some other standards. These standards are incorporated herein by reference. A system that implements the High Rate Packet Data specification of the cdma2000 standard is referred to herein as a high data rate (HDR) system. Proposed

[0009] There is therefore a need in the art for a finger front end capable of processing a large number of channels delivered at high chip rate in a high throughput, hardware efficient manner.

SUMMARY OF THE INVENTION

[0010] Embodiments disclosed herein address the need for increased finger demodulation capability in a hardware efficient manner. In one aspect, I and Q samples are shifted into a parallel-accessible shift register. A plurality of chip samples are accessed from the shift register and operated on in parallel to produce a multi-chip result for a channel each cycle. These multi-chip results can be accumulated and output to a symbol-rate processor on symbol boundaries. The scheduling of shift register access, computation, and accumulation can be scheduled such that the hardware is time-shared to support a large number of channels. In another aspect, time-tracking of a large number of channels can be accommodated through channel-specific indexing of the contents of the shift register file. These aspects, along with various others also presented, provide for hardware efficient chip rate processing capability for a large number of channels, with a high degree of flexibility in deployment of those channels.

[0011] The invention provides methods and system elements that implement various aspects, embodiments, and features of the invention, as described in further detail below.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The features, nature, and advantages of the present invention will become more apparent from the detailed description set forth below when taken in conjunction with the drawings in which like reference characters identify correspondingly throughout and wherein:

[0013] FIG. 1 is a wireless communication system that supports a number of users, and which can implement various aspects of the invention;

[0014] FIG. 2 depicts a CDMA receiver;

[0015] FIGS. 3A and 3B are two generalized embodiments of finger front ends configured in accordance with the present invention;

[0018] FIG. 5 is a block diagram of a finger front end with parameters specified, demonstrating various aspects of the present invention.

[0021] For clarity, the examples used in describing this invention will assume access points as the originator of pilot signals and access terminals as receivers and acquirers of those pilot signals, i.e. pilot signals on the forward link. Those skilled in the art would understand that access terminals as well as access points are can be equipped to transmit data with a pilot signal as described herein and the aspects of the present invention apply in those situations as well. The word “exemplary” is used

[0023] It is a common technique to dedicate specific hardware in a finger front end module to perform chip rate processing and to perform the relatively slower symbol rate processing in a DSP. Of course, those skilled in the art will recognize that the aspects of this invention apply with equal force if discrete hardware is deployed in lieu of DSP 240. Finger front ends typically are equipped to handle some number of channels or multipath signals simultaneously, and support for a greater number of channels provides increased performance. A common technique to increase channel support is to simply duplicate a single finger's hardware to provide support for as many fingers as desired. However, as the number of channels supported grows, the associated hardware can become prohibitively expensive. Finger front end 220 provides support for a large number of channels in a particularly hardware efficient manner.

[0027] FIG. 3A depicts a more detailed embodiment, finger front end 300. Finger front end 300 is one embodiment that could be deployed as finger front end 220 shown in FIG. 2 above. Finger front end 300 provides a hardware architecture that combines time-sharing and parallelism to support demodulation of multiple channels simultaneously. An unlimited number of configurations can be implemented based upon a number of parameter values, described in detail below.

[0028] The number of channels that can be simultaneously demodulated with this architecture is defined as parameter MAX_CHANNELS. MAX_CHANNELS is a function of two other parameters, S and P. S is the sampling rate, the rate at which I and Q samples are delivered to shift register file 350 (explained further below). P is a parallelism factor, determined by the number of chips of I and Q data that are processed per cycle. With no time tracking, MAX_CHANNELS would be determined as P*S. However, with time tracking, MAX_CHANNELS is determined as (P*S) - 2 to prevent overruns or underruns of shift register file 350.

[0029] In FIG. 3A, scheduler and timing control unit 310 (hereinafter scheduler 310) is shown connecting to the rest of the blocks in the figure either directly, or through another block. Scheduler 310 provides timing control for the various blocks as they are used to process the number of channels determined by MAX_CHANNELS. Each channel is processed sequentially, one channel per cycle, where a cycle is defined by the sampling rate. One common sampling rate employed in CDMA systems is 8 times the chip rate, commonly known as chipx8. Any sampling rate is supported by the current invention, however. During each cycle, the channel that is being processed is called the active channel. A round is defined as the processing of each channel once, in succession. The number of cycles needed to complete a round, CYCLES_PER_ROUND, is determined as MAX_CHANNELS + 1. This is so because each channel, of which there are MAX_CHANNELS, requires one cycle for computation, and an additional spare cycle is needed to allow time tracking to take place. The details of time tracking will be given with respect to the flowchart in FIG. 4 below.

[0030] I and Q samples are shifted into and stored in shift register file 350 at the sampling rate of S samples per chip. The data stored is addressable via an address labeled index in FIG. 3A, shown as an output from scheduler 310. Each cycle, the index provided to shift register file 350 corresponds to the currently active channel. For each access based on index, P pairs of I and Q data are retrieved from shift register file 350. This allows P chips, of I and Q data, to be demodulated simultaneously. The pairs retrieved are spaced S samples apart, as appropriate since P chips of data are desired. Maintaining S samples per chip in shift register file 350 allows time tracking to be performed via simple updating of the address given as

[illegible]

index, and allows early, late, or other data to be demodulated as well. The length of shift register file 350 must be sufficient to hold P chips worth of data, plus an additional amount of storage to buffer the data until it is used in a round without being shifted out and lost prematurely. The shifter length required, SHIFTER_LEN , can be determined as $\text{CYCLES_PER_ROUND} + (P-1)S$.

[0031] Despreader 360, rotator 370, deconvolver 380, and adder tree 390 make up block 355 which is referred to herein as a parallel sum. The P pairs of I and Q values from shift register file 350 are delivered to despreader 360, which contains P parallel despreaders for despreading them with P pairs of I and Q PN values delivered from PN generator 320. Despreading techniques are commonly known in the art. The P resultant despread I and Q pairs are delivered to rotator 370, where the I and Q pairs are rotated in P rotators according to the P outputs of phase generator 330. In the illustrative embodiment, the despread, rotated pairs are delivered to deconvolver 380, where P OVFSF codes are delivered from OVFSF generator 340 to deconvolve them. The deconvolved I values are then summed in adder tree 390 to produce a P chip I sum, and the deconvolved Q values are similarly summed in adder tree 390 to produce a P chip Q sum. The result of calculating a single I,Q result from P I,Q pairs is called a parallel sum. The parallel sum is calculated once per cycle, each cycle for one channel, until every channel is successively calculated. During idle cycles, a parallel sum need not be calculated, or the output of parallel sum 355 can simply be ignored. Rotators are optional – instead of frequency adjusting each signal independently, an overall frequency adjustment can be calculated and compensated for in clock generation circuitry (not shown). The present invention can be practiced in an alternate embodiment utilizing neither phase generator 330 nor rotator 370.

[0032] The parallel sum 355 output is delivered to accumulator 395 where it is added to a partial accumulation value corresponding to the active channel accessed in accumulator 395 (there is a separate accumulation for both the I and the Q for each active channel). Unless a symbol boundary has been reached, the new partial accumulation is stored in accumulator 395 in a location corresponding to the active channel. When a symbol boundary has been reached, the number of chips designated by the spreading factor, SF, for the active channel, have been accumulated in the partial sum. In this case, the I and Q accumulations correspond to the energy in the

[0034] PN generator 320 produces P pairs of I and Q PN data each cycle based on a value pn_count delivered from scheduler 310. There are a variety of types of PN sequences. For example, in IS-95 systems a single I and a single Q PN sequence which can be generated from linear feedback shift registers are used for spreading and despreading, with base stations identifying themselves via unique offsets in those PN sequences. On the other hand, in W-CDMA systems, the PN sequences are generated using Gold codes, and each base station identifies itself using a unique code. The aspects of this invention apply regardless of which type of PN sequence is used, or how PN generator 320 is implemented. Scheduler 310 keeps a PN count for each channel and provides the PN count for the active channel, denoted pn_count in FIG. 3A, to PN generator 320 for calculating the appropriate P pairs of I and Q PN values for despreading in despreader 360. Examples of PN generators useful in this context would include ROM based look up tables, indexed on pn_count, or one of the variety of masking schemes known in the art.

[0037] Those skilled in the art will recognize that these descriptions delineate blocks based on functionality for descriptive purposes only. One could redraw FIG. 3A with PN generator 320, phase generator 330, and OVFSF generator 340 subsumed into either the blocks that receive their respective outputs or into scheduler 310.

[0038] FIG. 3B depicts an alternate embodiment, finger front end 305. The discussion relating to FIG. 3A holds in its entirety for this figure, except with respect to the location of the rotator. Blocks that remain identical between the two figures are given like numbers in each figure. Scheduler 310 continues to drive shift register file 350, PN generator 320, OVSF generator 340, and accumulator 395. I and Q samples continue to arrive at shift register file 350. PN generator 320 and OVSF generator 340 drive despreader 360 and deconvolver 380, respectively. The change is made within parallel sum 356, which differs from parallel sum 355 of FIG. 3A. In FIG. 3A, rotator 370 was placed between despreader 360 and deconvolver 380. As such, it fell within the area of the circuit where P rotators were required to process the results of parallel sum 355. In FIG. 3B, the rotator, now numbered 375, is placed after the adder tree. Despreader 360 results are passed directly to deconvolver 380, then to adder tree 390, and finally into rotator 375. The benefit of placing rotator 375 after the adder tree is that only a single rotator is necessary, not P as in FIG. 3A. Scheduler 310 drives phase generator 335, which is designated differently than phase generator 330 since it only requires a single phase value per channel to be stored. This configuration provides an averaging effect that is less accurate than the chip-by-chip rotation of FIG. 3A, but may prove useful when hardware complexity is at a premium and this type of frequency compensation is sufficient.

[0039] As discussed above, the present invention provides a hardware efficient solution for providing support for demodulating a large number of channels simultaneously (MAX_CHANNELS, to be precise). The manner in which the support is provided also provides great flexibility for how the resources are allocated. For example, in prior art finger front ends which duplicated one finger's hardware M number of times, the ability to trade off resources was limited. Such a configuration would typically produce early, late and on-time data for M pilots and M data streams. As such, essentially 4M channels would be deployed, but a maximum of M data streams would result. In the present invention, the DSP is free to allocate the channel resources in a variety of ways. Like the older hardware versions, one option is to demodulate one pilot, a corresponding data signal, and an early and late stream for time tracking. In addition, however, a single pilot can be demodulated with a larger number of corresponding data streams, and only one early and late stream to provide

time tracking. This is useful when the transmitted signal bundles more than one data stream with unique codes and transmits them all with a common pilot.

[0040] FIG. 4 is a flowchart detailing how a scheduler, such as scheduler 310, can perform proper indexing, symbol boundary detection, and time tracking. Note that the subscript CH on a variable indicates that each individual channel has a unique variable of that name, and use of the variable indicates it is the variable corresponding to the active channel (contained in variable CH).

[0041] The flowchart operates as follows. Begin in block 400. For discussion, it is assumed that the active channel, CH, is initialized to zero, and all variables are initialized. In general, a DSP, such as DSP 240, is free to allocate a new channel by supplying the variables defining it. These include a spreading factor (SF), a PN offset (PN_OFFSET) to identify the PN sequence (either an offset in a common sequence or a unique sequence), and a covering code for that channel (OVSF_CODE). Note that, typically, pilot channels are not covered, so an all zeros OVSF_CODE can be assigned in those cases. The updating of variables for a particular channel is not shown in FIG. 4. The assumption is that the DSP is free to update channel parameters at will, and that appropriate safeguards will be taken to avoid overwriting a channel variable while it is active.

[0042] The distinction between an early or late channel is not of importance within this finger front end. The DSP can simply assign the time tracking channels by using the appropriate shift in the PN sequence and use the resultant symbols to perform time-track processing. All channels are treated uniformly by the finger front end.

[0043] Returning to the flowchart, from 400 proceed to 402. Check if $\text{index}_{\text{CH}} < 0$. The variable, index, is used generally as the address of the shift register file, where the most recent samples are stored in location 0 and the oldest remain in location (SHIFTER_LEN - 1). Index values of less than 0 are not valid addresses, so this is used to determine when to enter an idle state. These correspond to retard commands when the index previously pointed to location 0 or 1 in the shift register file, or on-time processing when the index pointed to 0. If $\text{index}_{\text{CH}} < 0$, proceed to 428, remain idle (update nothing, output nothing), then proceed to 430. In 430,

10052436 : 050904

increment index_{CH} by CYCLES_PER_ROUND . The cycle is finished. Proceed to 432 and increment CH by one to process the next channel.

[0044] From 432 proceed to 434 to check if $\text{CH} = \text{CYCLES_PER_ROUND}$. If so, then the round is over since CYCLES_PER_ROUND has been reached from a starting value of zero. Proceed to 436, remain idle (do no channel processing), and reset CH to zero. Proceed back to 434 where CH will not equal CYCLES_PER_ROUND since it has just been reset. Proceed back to 402 to check if $\text{index}_{\text{CH}} < 0$, as discussed above.

[0045] If index_{CH} is not less than zero, channel processing will commence. Proceed to 404 and access the shift register file using index_{CH} . Proceed to 406 and calculate $\text{parallel_sum}_{\text{CH}}$ (as described previously with respect to FIGS. 3A and 3B, and detailed in flow chart form in FIGS. 4A and 4B below). Proceed to 408 and accumulate $\text{parallel_sum}_{\text{CH}}$ by adding $\text{parallel_sum}_{\text{CH}}$ to accum_{CH} . Proceed to 410.

[0046] In 410, check if a symbol boundary for this channel has been reached. One method is to test if $\text{pn_count}_{\text{CH}} \% \text{SF}_{\text{CH}} = 0$, where $\text{pn_count}_{\text{CH}}$ is the current PN location for the active channel and SF_{CH} is its spreading factor. If not, proceed to 416. If so, a symbol boundary has been reached. Proceed to 412 and output accum_{CH} . Proceed to 414 and reset accum_{CH} to zero. Note that block 414 depicts the reset value as (0,0). This is to indicate that the accumulator is accumulating both an I and a Q value, so both need to be reset to zero. Proceed to 416.

[0047] In 416, check if an advance command has been given to this channel. If so, proceed to 422. If not, proceed to 418 to check if a retard command has been given. If so, proceed to 426 and decrement index_{CH} by two. Then proceed to 422. If a retard command was not issued, proceed to 420. In 420, decrement index_{CH} by one. Decrementing by one is the action taken when neither an advance nor a retard command is given. A retard causes an extra decrement to occur. An advance removes the decrement. Blocks 416, 418, 420, and 426 are the time-tracking blocks. As stated, when finished with an advance, retard, or on-time adjustment to index_{CH} , proceed to 422.

[0048] In 422, decrement index_{CH} by $(P*S)-1$. Proceed to 424 and update $\text{pn_count}_{\text{CH}}$ by incrementing by P. This is because P chips are processed each cycle. Proceed to 430, where, as described above, index_{CH} is incremented by

CYCLES_PER_ROUND. Then, in block 432, CH is incremented by one and the process repeats for the next channel in the round.

[0049] It will be clear to skilled artisans that some of the increment and decrement steps just described will collapse into fewer steps when the fixed parameters are set, as they will be in any particular implementation. The sequence of steps remains general and applies for any combination of P and S (from which the other parameters are derived).

[0050] A detail that is not shown is the treatment of non-assigned channels in this process. Regardless of whether or not all the channels are assigned and active, to maintain the proper timing, all channels plus the idle state are cycled through each round. There are a variety of ways to handle unassigned channels. A power efficient method would be to leave all the signals that ultimately cause computation in the parallel sum to remain unchanged, and thus excess toggling of the hardware is reduced. Similarly, the accumulator can be disabled when processing an unassigned channel. The accumulator output can be turned off for an unassigned channel. Or, the DSP (or other symbol rate processor) can simply ignore results generated for unassigned channels.

[0051] FIG. 4A depicts a detailed embodiment of step 406, calculating the parallel sum. This procedure corresponds to the apparatus depicted in FIG. 3A above. In 440A, supply pn_count_{CH} to the PN generator. Despread the output of the shift register file with the output of the PN generator. Proceed to 442A. Supply Δ_{CH} to the phase generator. Rotate despread results with phase generator output. As in FIG. 3A, this rotator requires P rotation computations or elements. Proceed to 444A. Supply pn_count_{CH} to OVSF generator. Discover the rotator results with OVSF generator output. Proceed to 446A and sum the discovered results.

[0052] FIG. 4B depicts an alternative embodiment of step 406, calculating the parallel sum. As in FIG. 3B, placing the rotator at the end of the process instead of between despreading and deconvolving lowers the rotation computations or elements from P to one. In 440B, supply pn_count_{CH} to the PN generator. Despread the output of the shift register file with the output of the PN generator. Proceed to 444B. Supply pn_count_{CH} to OVFSF generator. Discover the despread results with OVFSF generator output. Proceed to 446B and sum the discovered results. Proceed to 442A. Supply

[illegible]

delta_{CH} to the phase generator. Rotate summed results with phase generator output. As in FIG. 3B, this rotator requires only one rotation computation or element.

[0053] FIG. 5 depicts an alternate embodiment labeled finger front end 500. Finger front end 500 is one embodiment that could be deployed as finger front end 220, described with respect to FIG. 2 above. Actual parameters will be chosen for this example, and the principle of handling spreading factors less than parallelism P will be further detailed through this example. A common sampling rate in CDMA receivers is at $\text{chip} \times 8$, and in this example S is set to 8. The level of parallelism supported, P, will also be set to 8. Hence, $\text{MAX_CHANNELS} = P \times S - 2 = 62$. $\text{CYCLES_PER_ROUND} = \text{MAX_CHANNELS} + 1 = 63$. $\text{SHIFTER_LEN} = \text{CYCLES_PER_ROUND} + (P-1)S = 119$. In this example, spreading factors as low as 4 are supported, as well as 8 and higher integer multiples of 8.

[0054] In FIG. 5, I and Q samples are shifted into and stored in shift register file 350 at the sampling rate of S samples per chip. The data stored is addressable via an address labeled index, shown as an output from scheduler and timing control unit 510 (hereinafter scheduler 510). Each cycle, the index provided to shift register file 550 corresponds to the currently active channel. For each access based on index, 8 pairs of I and Q data are retrieved from shift register file 550. This allows 8 chips of I and Q data to be demodulated simultaneously. The pairs retrieved are spaced 8 samples apart, as appropriate since 8 chips of data are desired.

[0055] The 8 pairs of I and Q values from shift register file 550 are delivered to despreader 560, which contains 8 parallel despreaders for despreading the 8 pairs of I and Q data with 8 pairs of I and Q PN values delivered from PN generator 520. The 8 resultant despread I and Q pairs are delivered to rotator 570, where they are rotated in 8 rotators according to the 8 outputs of phase generator 530. The despread, rotated pairs are delivered to decoder 580, where 8 OVFSF codes are delivered from OVFSF generator 840 to decode them. The first 4 of the 8 decoded I values are then summed in adder tree 590 to produce a 4 chip I sum, and the first 4 of the 8 decoded Q values are similarly summed in adder tree 590 to produce a 4 chip Q sum. The second 4 of the 8 decoded I values are summed in adder tree 592 to produce a second 4 chip I sum, and the second 4 of the 8 decoded Q values are similarly summed in adder tree 592 to produce a second 4 chip Q sum.

[0056] The resultant first and second 4 chip I and Q sums from adder trees 590 and 592, respectively, are delivered to final adder stage 594 to produce an 8 chip I sum and an 8 chip Q sum. The first and second 4 chip I and Q sums from adder trees 590 and 592, respectively, are also delivered to multiplexor 596. When the spreading factor (SF) of the active channel is 4, there are two symbols completed during the single chipx8 cycle. Multiplexor 596 is directed by scheduler 510 to deliver the two symbols of I and Q data to the symbol rate processor (not shown).

[0057] The output of final adder stage 594 is added in adder 598 with partial accumulation for the active channel stored in partial accum RAM 599. Final adder stage 598 and partial accum RAM make up the accumulator function, which is controlled by scheduler 510 to output the results through multiplexor 596 for delivery to the symbol rate processor at symbol boundaries. For SF not equal to 4, the output of partial accum RAM is selected in multiplexor 596. Scheduler 510 also controls the resetting of the active channel partial accumulation value. Naturally, when SF=8, there is no actual accumulation needed since an 8 chip result is calculated in final adder stage 598. When SF=8 the partial accumulation is constantly set to zero and the 8 chip result is delivered to multiplexor 596. (An alternate, not shown, is to have multiplexor 596 take the output of final adder stage 594 as an input and an additional select line to deliver is when SF=8). For spreading factors greater than 8, accumulation occurs in a similar fashion as described with respect to FIG. 3A. As before, there is a separate accumulation for both the I and the Q results for each active channel. Unless a symbol boundary has been reached, the new partial accumulation, calculated in adder 598, is stored in partial accum RAM 599 in a location corresponding to the active channel. Again, If a symbol boundary has been reached, meaning the number of chips designated by the spreading factor, SF, for the active channel, have been accumulated, then the I and Q accumulations correspond to the energy in the symbols and are delivered to the symbol rate processor. The partial accumulation values then stored in partial accum RAM 599 for the active channel will be reset to zero, under control of scheduler 510. Scheduler 510 maintains a spreading factor (SF) value for each channel and determines when the symbol boundary has been reached.

005243-050601

[0058] The previous paragraph has detailed one possible configuration supporting spreading factors smaller than the parallelism deployed. In general, for larger values of P and/or smaller values of SF, the appropriate taps can be added earlier in the adder tree to extract symbol data. Those earlier taps can be multiplexed in the fashion described to deliver the symbol data to the symbol rate processor.

[0059] The discussion of FIG. 3A above relating to PN generator 320 , phase generator 330, and OVSF generator 340 applies to PN generator 520, phase generator 530, and OVSF generator 540, respectively, in FIG. 5. Naturally, scheduler 510 replaces scheduler 310 when making that translation.

[0060] The same principle of rotator location discussed in the contrast between FIGS. 3A and 3B applies to the embodiment depicted in FIG. 5. The details of the second option are not shown, but will be clear to those of skill in the art.

[0061] The flowchart of FIG. 4 is suitable to describe the functioning of scheduler 510 and its interrelationship with the various blocks of FIG. 5. Clearly the generalized parameters in FIG. 4 will now have numerical values inserted, i.e. CYCLES_PER_ROUND is 63, $P = 8$, and $S = 8$. Calculation of the parallel sum described in step 406 will encompass the additional tap values created by breaking a single adder tree into adder trees 590 and 592 and final adder stage 594 (the parallel sum value is the output of final adder 594). Symbol boundary output step 412 will encompass the multiplexing of the additional tap values for SF values less than P (i.e. $SF = 4$ and $P = 8$) for output to a symbol rate processor. Aside from these refinements, the processing flow for cycling through channels in the round, accumulation, updating pn_count_{CH} , and updating $index_{CH}$ (including time tracking) remains the same.

[0062] It should be noted that in all the embodiments described above, method steps can be interchanged without departing from the scope of the invention.

[0063] Those of skill in the art will understand that information and signals may be represented using any of a variety of different technologies and techniques. For example, data, instructions, commands, information, signals, bits, symbols, and chips that may be referenced throughout the above description may be represented by voltages, currents, electromagnetic waves, magnetic fields or particles, optical fields or particles, or any combination thereof.

[illegible]

[0066] The steps of a method or algorithm described in connection with the embodiments disclosed herein may be embodied directly in hardware, in a software module executed by a processor, or in a combination of the two. A software module may reside in RAM memory, flash memory, ROM memory, EPROM memory, EEPROM memory, registers, hard disk, a removable disk, a CD-ROM, or any other form of storage medium known in the art. An exemplary storage medium is coupled to the processor such the processor can read information from, and write information to, the storage medium. In the alternative, the storage medium may be integral to the processor. The processor and the storage medium may reside in an ASIC. The ASIC

may reside in a user terminal. In the alternative, the processor and the storage medium may reside as discrete components in a user terminal.

[0067] The previous description of the disclosed embodiments is provided to enable any person skilled in the art to make or use the present invention. Various modifications to these embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments without departing from the spirit or scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown herein but is to be accorded the widest scope consistent with the principles and novel features disclosed herein.

005436 05001
"06050" 9E25860